NeoSOFT Database Guidelines

For

MS DOT NET Department

V1.2

Document Revision History

Ver. No.	Date	Change Information	Author	Reviewed/ Approved By	Reviewed/ Approved Date	Release Date
1.0		First draft of the training material	Vaibhav Kulkarni	Bhavesh Patel		
1.1		Segregation in sections	Vaibhav Kulkarni	Bhavesh Patel		
1.2	27-Oct-16	Removal of redundant section	Vaibhav Kulkarni	Bhavesh Patel		27-Oct-16

Template Revision History

Ver. No.	Date	Change Information	Author	Reviewed/ Approved By	Reviewed/ Approved Date	Release Date
1.0						

Table of Contents

Overview

This document contains the naming conventions and guidelines to follow in the database design and database programming. They're the base for scalability, maintenance, re usability and availability of data.

Naming Conventions

A naming convention is an important part of well-built data model. The main goal of adopting and following naming convention for database object is, everyone can easily find and identify the purpose of all objects contained in database.

Naming convention details mentioned in following section can be used with any type of database, may it be MS SQL, MySQL, Oracle etc.

Following 7 types of objects contribute to any database, therefore naming conventions are described for each database object.

1. Tables
2. Columns
3. Index
4. Constraints
5. Views
6. Stored Procedures
7. Functions

General Conventions

1. Use Pascal notations for database objects, example CustomerCategory.
2. Every object name should be meaningful and easily understandable to all.
3. Avoid using numbers in any object name.
4. Every database object name should start with an alphabet.
5. Limit object name up to maximum 50 characters.
6. Use of "_" should be avoided in table names and can be used only for other objects as specified in following sections.
7. Do not use spaces in the name of database objects.
8. Do not use SQL keywords as the name of database object name.
9. Do not use any abbreviation for database object name. It may lead to misinterpretation.
10. Do not use any special characters in database object name.

Tables

1. Table Name should be **Singular** as it represents an entity.

   Good – Product

   Not Good – Products
2. Do not use any prefix for table name.
3. "_" should not be used anywhere in table name.
4. Do not use any white space in table name even though it is supported by some of the databases.

   Good – ProductCategory

   Not Good – [Product Category]
5. **Junction Table -** Junction table or mapping table should have proper name which should be combination of individual table names.

   Example - If there are Doctor and Patient table, there is a possibility to have many to many mapping between both of them. In such cases, mapping table should be named as **DoctorPatient**.
6. Each table should have primary key, preferably auto identity column.
7. Auto identity columns should not be required in mapping/junction tables.

Columns

1. Pascal Casing should be used for naming column names.
2. Do not use any prefix for column names.
3. Field name should not be longer than 50 characters.
4. No need to repeat table name for any column name except identifier field.
5. Identifier column should be named as **<TableName> + "Id"** example, **ProductId**
6. Foreign Key field should have same name as Primary key field name in its original table.

   Example

   In **Category** table, there will be a primary key as **CategoryId**.

   In this case, **Product** table must have a foreign key field with same name as **CategoryId**
7. In case of multiple foreign keys to the same primary key field in another table, it should be prefixed with some valid descriptor.

   Example

   If there is an Address table, there can be another table with foreign keys like HomeAddress, WorkAddress.

Indexes

1. By default, Clustered index should be created on Primary Key field.
2. Index name should use the following pattern,

   IX_<Table Name>_<Field/Column Name>

Constraints

1. Constraints are at the field/column level so the name of field should be used in a constraint.
2. Type of constraint (Check, Referential Integrity/Foreign Key, Primary Key, Unique) should be prefixed with valid abbreviation.
3. Constraint name should use the following pattern,

   <Constraint Type>_<Table Name>_<Field Name>

   Examples

   a. PK_Product_ProductId

   b. FK_Order_ProductId
4. Following prefixes to be used for <Constraint Type> in above naming pattern

   a. Primary Key – PK

   b. Foreign Key – FK

   c. Check – CK

   d. Unique – UN

Views

1. View name should be prefixed with **VW_<Table Name>.**
2. If View is using joins on two or multiple tables, all such tables should be part of view name for clear identification.

   Example - VW_Category_Product

Stored Procedures

1. Stored Procedure name should be prefixed with **STP_<Procedure name>**
2. Procedure name should follow Pascal Casing.
3. SQL Server uses **SP_** as a prefix for system defined stored procedures, so same should not be used for any user defined stored procedures.
4. Following operation names should be used in procedure name.
    a. Insert - example – **STP_InsertProduct**
    b. Update - example – **STP_UpdateProduct**
    c. Delete - example – **STP_DeleteProduct**
    d. Get<Entity name>ById - example – **STP_GetProductById**
    e. Get<Entity name>List – example – **STP_GetProductList**
5. All input and output parameters for stored procedure should follow Pascal Casing notation.

Functions

1. Function should be prefixed with **FN_<Function name>**
2. Function name should follow Pascal Casing.
3. Function name should be verb because it will always return a value. Example – FN_GetFormattedDate().
4. Other conventions for function should be same as stored procedures.

Variables

1. All variables within function or stored procedure should follow camel casing.
   Example - @firstName, @result, @initialValue etc.
2. Each variable should be prefixed with **@.**
3. Do not declare any variable in the loop. Declare it outside and then use it in the loop.

Database Designing Guidelines

A strong database design is a backbone of any system, therefore adequate attention should be provided to database design. Following points to be considered for database designing,

1. Database should be normalized up to min 3rd normal form.
2. Database normalization can be checked in detail in the following links,
   http://www.studytonight.com/dbms/database-normalization.php
   http://www.w3schools.in/dbms/database-normalization/
3. Over normalization should be avoided as it may make the query structure complex.
4. Each table created within the database should have a Primary Key and a Clustered Index.
5. GUID fields should **not** be used for clustered index even if used as table's Primary Key.
6. Composite keys should be avoided. Surrogate INT IDENTITY keys can be used as a replacement.
7. Columns, set to the smallest size possible – where ever possible avoiding NVARCHAR (MAX), TEXT etc. data types. Use the maximum allowed characters of VARCHAR instead.
8. Avoid Nullable columns in tables. Nullable columns require extra space within the database to designate the fact that they are nullable. Set an empty string instead nullable columns.
9. Use referential integrity – Foreign keys and unique constraints should be applied.
10. Columns with default values should not allow NULL.
11. The field's size of any field will be the smallest size capable to hold the highest or longest value for the field.
12. The table's declaration should include all the elements (indexes, foreign keys, constraints, default values, etc.) necessary to represent the nature of data saved on the table.
13. Each table attributed for Audit Trail should have following fields mandatorily,
    a. CreatedOn
    b. CreatedBy
    c. UpdatedOn
    d. UpdatedBy
    e. DeletedOn
    f. DeletedBy
14. Wherever possible, always use Soft Delete option instead of Hard Delete based on business requirement.

Code Formatting

1. Use upper case for all SQL keywords (Example: SELECT, INSERT, UPDATE, WHERE, AND, OR, LIKE, etc.)
2. Indent code to improve readability.
3. Comment code blocks that are not easily understandable
    a. Use single-line comment markers (--).
    b. Reserve multi-line comments (/*.. ..*/) for blocking out sections of code.
4. Every stored procedure and function should have following comment structure

```
/* ------------------------------------------------------------------------------
   Name –
   Purpose –
   Author –
   Created On –
   Updated On –
   Test Execute Statement – STP_ValidateUser('test','test')
------------------------------------------------------------------------------*/
```

   Note – Be cautious while writing an Author Name. Consult your team leader while writing Author Name.

5. Use parentheses to increase readability (Example : WHERE (color='red' AND (size = 1 OR size = 2)))
6. Use BEGIN..END blocks only when multiple statements are present within a conditional code segment.
7. Use one blank line to separate code sections.
8. Use spaces so that expressions read like sentences (Example : fillfactor = 25, not fillfactor=25)
9. Format JOIN operations using indents for better readability.

Updating database content

1. All the content updates (Insert, Update, Delete statements) should be done through parameterized stored procedures, to avoid **SQL injection**.
2. Every Insert related stored procedure should return a complete record from table (with inserted Id).
3. Always use transactions with Try Catch block when performing (Insert, Update, Delete) operations with related/nested tables.
4. Do not use transaction statement for single (Insert, Update, Delete) table.
5. Do not use Transactions with Select statements.
6. Always use transaction carefully, unwanted transaction can reduce performance of execution.
7. The actions will be executed and logged in such way that a full auditing review is possible.
8. Always use **Where** condition while performing Update or Delete request.
9. Implement persistence mechanism for documents data. A relational database has the advantage to enforce and protect the references among tables, a problem might occur when changing an entry in one table (used in relations) affects the information that it's necessary to keep static over the time in other tables. One example could be updating the name of a user who generated some documents in the past, it's necessary to keep the past documents intact over the time. A possible solution for this case is one xml field at the document level who keeps "the snapshot" of the document and this document representation is going to be used when the information needs to be invariable over the time. This solution is better than adding multiple fields on the tables, or keeping log tables where the information is not directly attached to the entity.

Requesting database content / Data Retrieval

1. All the content requests (select statements) should be done through parameterized stored procedures, functions and/or views (optionally, if view exist).
2. Stored procedures should be used as much as possible, to allow query execution plans to be re-used
3. Do not use SELECT *, instead SELECT only the columns needed to keep the query's memory usage as low as possible
4. Where ever possible avoid cursors and use sql built in events/functions.
5. SET NOCOUNT ON should be at the start of each SQL batch to reduce network traffic
6. Dynamic SQL should be executed using **sp_executesql**.
7. Do not repeatedly call functions within stored procedures, functions, batches and triggers
8. Confirm queries executed are able to seek on indexes in database
9. Where ever possible Avoid wildcard characters at the beginning of a word while searching using the LIKE keyword
10. Where ever possible Avoid searching using not equals operators (<> and NOT)
11. Where ever possible Avoid functions in WHERE clause on table columns
12. Where ever possible Avoid implicit conversions in WHERE clause (use CAST/CONVERT if necessary)
13. The select statements will contain explicitly the names of the columns that will be returned.
14. The result sets should be optimized to reduce the communication between client applications or servers to the database server. It means that the data requests should return the minimum of columns.
15. Avoid the use of temporally tables, use (CTE) common table expressions instead.

Code Programming

1. The code should contain enough comments to understand the programming logic implemented on each case.
2. Avoid the use of cursors when the operations can be executed in SQL batches.
3. Use column names on the queries; never make references to column positions.
4. User joins instead of where/and constraints on the queries
5. Use the char type when the column is not nullable and the amount of characters always will be the same, in other cases use varchar instead. Keeping the minimum size per register is a paramount.
6. User varchar instead of text wherever possible.
7. The stored procedures should include a parameter @debug. In debug mode the intermediate results can be printed.
8. Optimize the amount of operations per stored procedure; local variables can be used to save temporary data instead of repeating the same function call multiple times on the stored procedure/function, etc.
9. The stored procedures should return the status, a way to check that the whole operation took place without any issue. Create a group of possible answers and reuse them for all the stored procedures.
10. Declare SET NOCOUNT ON at the beginning of the SQL batches.
11. Try to avoid storing binary data on the database, keep them in physical files and create fields on the tables to reference them.
12. Avoid the use of dynamic SQL queries as much as possible.
13. Keep transactions as short as possible, they lock resources and if it isn't implemented properly, it may affect the performance of concurrent operations.
14. Check the variable @@ERROR after each update operation.
15. Make sure the user receives a friendly error message whenever the SQL operation failed.
16. Use a consistent use of date format in all the tables, 4 digits for year in all cases.
17. Where ever possible always use "user defined table type (data table)" or XML for inserting multiple records (Bulk insert).
18. **MERGE** statement should be used for bulk insert/update/delete operations.
19. Where ever possible avoid triggers.
20. Where ever possible use Join instead of Sub queries.
21. Keep passwords as encrypted for security. Use of Salt mechanism is recommended.
22. Choose columns with the integer data type (or its variants) for indexing. varchar column indexing will cause performance problems.
23. Where ever possible for better SELECT performance, index your JOIN columns.